Prosjekt Ymer, grafikkprosessering på sky

Linda Granstad
Høgskolen i Oslo, ingeniørutdanningen
Avdeling for datateknologi
linda.granstad@gmail.com

Kyrre Begnum
Høgskolen i Oslo, ingeniørutdanningen
Avdeling for datateknologi
kyrre.begnum@iu.hio.no

Sammendrag

Skyteknologi er en teknologi som tilbyr et spekter av nye tjenester som er lite utprøvd på mange felt. Et av disse områdene er bruk av sky i forbindelse med grafikkprosessering. Vil det være mulig å få en renderfarm til å fungere på en sky, og vil det fungere bra nok til å kunne bli en reell løsning for små og mellomstore bedrifter, i forhold til kvalitet, ytelse og lønnsomhet?

Ved Høgskolen i Oslo, avdeling for ingeniørutdanning, ble Prosjekt Ymer gjennomført våren 2010, som et studentprosjekt. Prosjektet bestod av implementering av prototype, samt testing av denne. Det ble også foretatt grundige kostnadsundersøkelser. Resultatene av arbeidet viser at det er mulig å kjøre en renderfarm på en sky, og det fungerer svært forutsigbart, men at det fortsatt er enkelte utfordringer. Fra et brukerperspektiv kan man konkludere med at det ville vært en svært utfordrende og kompleks oppgave å skape en løsning som Ymer for personer uten inngående IT-kunnskaper. De fleste komponentene som ble brukt i dette prosjektet eksisterer fra før, men det er svært mangelfull informasjon på hvordan disse settes sammen, og hvordan man løser problematikken som eksempelvis dynamisk DNS medfører. Ymer ligger nå veldokumentert på internett, tilgjengelig for alle, og det vil kreve langt mindre ferdigheter av brukeren å ta denne i bruk, sammenlignet med hva som ville vært påkrevd om brukeren hadde ønsket forsøke å lage noe eget. Prosjekt Ymer viser at det er både mulig og lønnsomt å benytte seg av alternative løsninger for å prosessere grafikk.

Innledning

Ved Høgskolen i Oslo, avdeling for ingeniørutdanning, ble Prosjekt Ymer gjennomført våren 2010, som ett studentprosjekt, hvorpå forfatterne av denne artikkelen var prosjektleder (Linda Granstad) og veileder (Kyrre Begnum). Prosjektet bestod av implementering av prototypen Ymer, samt testing av denne. Det ble også foretatt grundige kostnadsundersøkelser for å undersøke om løsningen ville være økonomisk bærekraftig.

Skybasert teknologi, såkalt ”cloud computing”, har potensial for å kunne bli en framtidig løsning for både miljø og økonomibesparelser. Teknologien bygger på grid-computing, det vil si ideen om å benytte seg av desentraliserte ressurser [1,2]. Grunntanken bak dette prinsippet er å gi forbrukeren datakraft, uten behov for selv å investere i dyr maskinvare. Dette har vært spesielt populært i miljøer med vitenskaplig tungregning. Et viktig argument for å benytte seg av skyteknologi er at en slik løsning ikke krever drifting fra forbrukerens side, eller innkjøp av kraftig maskinvare. Siden en sky i praksis kan levere det forbrukeren etterspør av kraft og består av forskjellige API (application programming interface), så vil man også kunne imøtekomme varierende behov, hvilket gjør skyen fleksibel og dynamisk.

Man kan anta at nesten all grafisk produksjon generert i verden i dag utføres ved hjelp av datamaskiner. Når man starter opp en mediebedrift vil man derfor måtte handle

This paper was presented at the NIK 2010 conference. For more information, see http://www.nik.no/

inn maskinvare for store summer. Dette gjør at de mindre bedriftene løper en høyere økonomisk risiko, og at de risikerer å tape mye marked til større konkurrenter. For bedrifter som møter på disse utfordringene, kan leie av ressurser fra en skytjeneste, være en løsning. Skytjenester har blitt en mer etablert måte å anskaffe datakraft på etter at virtualiseringsteknologier de siste årene har fått stadig bedre ytelse [3]. Her finnes det flere forskjellige leverandører som leverer et spekter av tjenester, men den aktøren som antas å være lengst fremme ved å kunne tilby et relativt bredt spekter av tjenester, er Amazon Web Services [4]. Flere prosjekter undersøker nå forskjellige måter hvordan skyteknologi kan utnyttes i større grad [5,6,7]. Fokus er da oftest på ytelse og fleksibilitet, men ikke nødvendigvis på brukeraspekter, slik som den høye tekniske terskelen for å ta distribuerte løsninger i bruk.

Vi ser for oss en mindre mediebedrift som for eksempel produserer kortere instruksjonsfilmer. En slik bedrift er i utgangspunktet ikke noen IT-bedrift hvor man innehar en spisskompetanse på datateknologi, og dermed vil også ressursene som tilbys i form av skyløsninger ikke være like lett tilgjengelig for denne brukergruppen. Man kan si at forbrukeren i dag tilbys verktøyet for å få en renderfarm til å fungere på sky, men at det vil kreve inngående IT-kompetanse for å sette sammen komponentene. Hensikten med dette prosjektet var å undersøke om det var teknisk mulig å lage en prototype av en renderfarm som kan kjøres på sky, som reduserer den tekniske vanskelighetsgraden for å realisere en slik løsning, samt å finne ut om en skytjeneste kan være et reelt alternativ for mindre bedrifter, som har behov for grafikkprosesseringsverktøy med fokus på ytelse, kvalitet og lønnsomhet.

Bakgrunn

Amazon EC2

Amazon EC2 er en skytjeneste levert av selskapet Amazon Web Services. Tjenesten tilbyr sine kunder mulighet til å leie maskinkraft i form av virtuelle maskiner på sky. Amazon baserer seg på Xen virtualiseringsteknologi, hvilket gjør skyen fleksibel, og i teorien kan kunden gjenskape et hvilket som helst datasystem de ønsker. Prisene Amazon EC2 opererer med følger prinsippet ”betal for det du bruker”. Timeprisen er den mest dominerende faktoren i totalkostnaden. Timeprisen varierer i forhold til mengde datakraft som brukes, noe som gjør skyen svært kostnadseffektiv dersom man er i stand til å kjøre de virtuelle instansene når de virkelig skal brukes.

Instanstyper

Amazon EC2 operer med instanstyper, noe som gjør det enkelt å skalere maskinkraft etter behov. I figur 1 vises en oversikt over hvilke instanstyper EC2 tilbyr sine kunder.

Instanstype:	Størrelse:	Minne:	Ec2 kjerner:
m1.small	Liten	1,7 GB	1
m1.large	Stor	7,5 GB	4
m1.xlarge	Ekstra stor	15 GB	8
m2.xlarge	Ekstra stor	17,1 GB	6,5
m2.2xlarge	Dobbel ekstra stor	34,2 GB	13
m2.4xlarge	Kvadruppel	68,4 GB	26
c1.medium	Medium	1,7 GB	5
c1.xlarge	Ekstra stor	7 GB	20

Figur 1 Amazon EC2 instanstyper. En EC2 kjerne er en virtuell prosessorkjerne.

Pris per time varierer i forhold til instanstype og region. Med region, så mener man hvor skyen er lokalisert. Figur 2 viser en oversikt over prisene, i forhold til region og instanstype.

Instanstype:	US East	US West	EU West	APAC Singapore
m1.small	$ 0.085 per time	$ 0.095 per time	$ 0.095 per time	$ 0.095 per time
m1.large	$ 0.34 per time	$ 0.38 per time	$ 0.38 per time	$ 0.38 per time
m1.xlarge	$ 0.68 per time	$ 0.76 per time	$ 0.76 per time	$ 0.76 per time
m2.xlarge	$ 0.50 per time	$ 0.57 per time	$ 0.57 per time	$ 0.57 per time
m2.2xlarge	$ 1.20 per time	$ 1.34 per time	$ 1.34 per time	$ 1.34 per time
m2.4xlarge	$ 2.40 per time	$ 2.68 per time	$ 2.68 per time	$ 2.68 per time
c1.medium	$ 0.17 per time	$ 0.19 per time	$ 0.19 per time	$ 0.19 per time
c1.xlarge	$ 0.68 per time	$ 0.76 per time	$ 0.76 per time	$ 0.76 per time

Figur 2 Amazon EC2 prisliste for region og instanstype per september 2010

Nettverk
Amazon EC2 benytter DHCP (Dynamic Host Configuration Protocol) til tildeling av IP-adresser. Dette betyr i praksis at det ikke er mulig og på forhånd kunne vite hvilken IP-adresse instansen vil få, hvilket kompliserer bruken av cluster på sky.

AWS management console
Amazon EC2 tilbyr i dag et enkelt brukergrensesnitt de kaller AWS management console. Dette brukergrensesnittet gjør det mulig å administrere instanser på skyen, samt andre tjenester som tilbys av Amazon Web Services. Grensesnittet tilbyr ingen mulighet til å gruppere sammen virtuelle maskiner som logisk sett høres sammen. Man kan altså ikke starte og stoppe en gruppe av virtuelle maskiner som en atomisk enhet.

Manage Large Networks (MLN)
MLN [8] er et verktøy utviklet for håndtering av grupper med virtuelle maskiner. MLN installeres på en Linux maskin og operasjonene utføres på kommandolinjen. MLN muliggjør konfigurasjon og administrasjon av grupper av virtuelle maskiner. Konfigurasjonen av de virtuelle maskinene blir spesifisert i et eget deklarativt språk, og blir spesifisert for prosjekter som inneholder en eller flere virtuelle maskiner. MLN støtter konfigurasjon av de vanligste system aspektene, slik som nettverksinnstillinger og brukere. Prosjektet kan så startes og stoppes som en enhet. Et pluginrammeverk gjør det mulig å utvide funksjonaliteten til MLN. Et eksempel på et slikt plugin er EC2 plugin, som gjør det mulig å administrere virtuelle maskiner lokalt på en server så vel som i Amazon AWS skyen [9].

Prototypen

Arkitektur

En renderfarm fungerer som en klassisk sentralisert clusterarkitektur med to nodetyper. Mesternoden styrer arbeidsfordeling på slavenodene som deretter utfører selve arbeidet. For å kunne sette opp en renderfarm som skal kunne kjøre på sky, må man sette opp en del komponenter. Dette er:

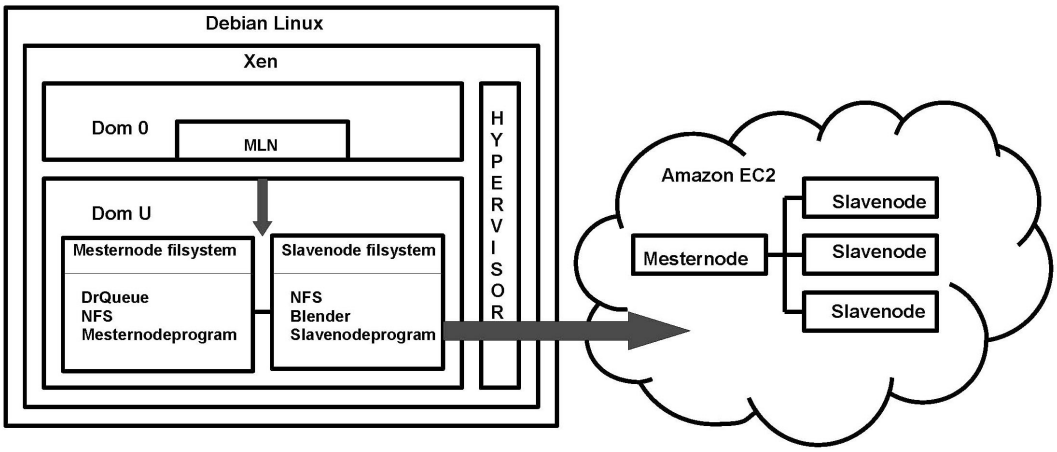
- Server
- Programvare for server/operativsystem
- Programvare for virtualisering
- Delt lagring
- IP-adresse eller vertsnavn
- Grafikkprosesseringsprogram
- Administrasjonsprogram for virtuelle nettverk
- Køhåndteringsprogram
- Skyleverandør
- Program som kobler nodene sammen

Implementering

Utgangspunktet for implementeringen av prototypen Ymer var en ordinær server med Debian Linux og Xen hypervisor [10], installert. Det ble bygget to filsystemer, som senere ble brukt som mal for mester og slavenode. Ferdiglagde filsystemer med forhåndsinstallert programvare er en vanlig fremgangsmåte når man vil håndterer store menger virtuelle maskiner [11]. I disse ble det installert nødvendig programvare, som NFS som delt lagring [12], DrQueue som køhåndteringsprogram [13] og Blender [14] til selve grfikkprossesering.

For å sørge for at nodene finner hverandre på skyen ble det skrevet et script i Perl for hver av nodetypene. Scriptene er komplekse, blant annet fordi Amazon EC2 tildeler nodene nye interne og offentlige IP-adresser i det de startes opp, og dermed vil de i praksis være ”blinde” på dette tidspunktet. Dette betyr at de ikke vil kunne vite hva deres egen IP og nettmaske er, og heller ikke hva IP-adressen til de andre nodene er. Fordi det heller ikke er mulig å forhåndskonfigurere det som er nødvendig for at NFS og DrQueue skal kunne fungere, sørger scriptene også for at mesternoden utveksler IP-adresse og nettmaske med slavenodene, slik at mesternoden kan konfigurere NFS og deretter restarte den. Først etter dette er det mulig for slavenodene å koble seg til mesternoden. DrQueue konfigureres i slavenodescriptet etter at den har koblet seg til, for så å starte opp DrQueue på alle noder.

For å forenkle implementering av Ymer ble MLN brukt. MLN har støtte for Xen virtualiseringsplattform, og har blant annet et plugin som gjør det mulig å sende klynger med virtuelle maskiner til Amazon EC2, samt et plugin som gjør det mulig å tilknytte mesternodens DynDNS, og dermed gjøre den mulig å spore i et privat nettverk. Med MLN definerer man hvordan renderfarmen skal se ut, ut i fra en konfigurasjonsfil. Etter at denne er ferdigdefinert, kan man enkelt bygge og starte renderfarmen med enkle kommandoer. Etter oppstart av renderfarmen på sky, kan man logge på mesternoden med DynDNS-adressen. Figur 3 viser Ymers arkitektur, og illustrerer hvordan man skyver rendernodene ut på skyen, ut fra den lokale serveren.



Figur 3 Ymers arkitektur

Tester og undersøkelser

Testene og undersøkelsene som har blitt uført i prosjektet, var ment å dekke en rekke aspekter ved bruk av Amazon EC2. Vi ser for oss en bedrift som vil bygge sin egen renderfarm ved hjelp av MLN. Dernest vil man starte skyen og må vente til alle noden har kommet opp og skyen har konfigurert seg selv før man kan sende jobber dit. Når en renderjobb er fullført, vil man så måtte laste ned den ferdige filen (filmen). Følgende hovedtester ble utført:

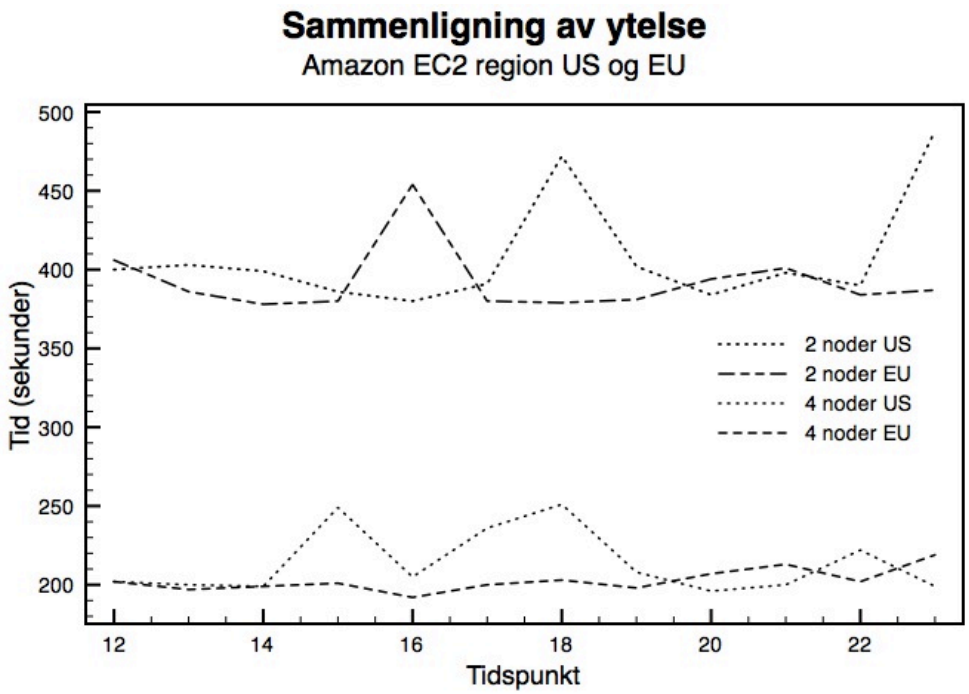
1. Måle tiden og døgnvariasjon på en grafikkprosesseringsoppgave på EC2, sone US, instanstype c1.small
2. Måle tiden og døgnvariasjon på en grafikkprosesseringsoppgave på EC2, sone EU, instanstype c1.small
3. Måle tiden det tar å bygge, sende, starte og pinge en renderfarm ut på en sky
4. Grafikkprosessering av en stor fil, med økt antall noder. Er det samsvar mellom økt antall noder og kraft brukt i forhold til tid? (proporsjonalt/uproporsjonalt?)
5. Måle nedlastningstid av stor fil fra sky, til server over 24 timer.
6. Måling av pakke og bytestrøm for NFS og DrQueue, under grafikkprosessering
7. Tid det tar å starte et prosjekt til alle Amazon har allokert minne til alle noder
8. Kostnader (undersøkelser og vurderinger rundt kostnadsaspektet)

De tekniske testene ble gjennomført delvis manuelt, og delvis gjennom script som kjørte i angitte intervaller.

Resultater

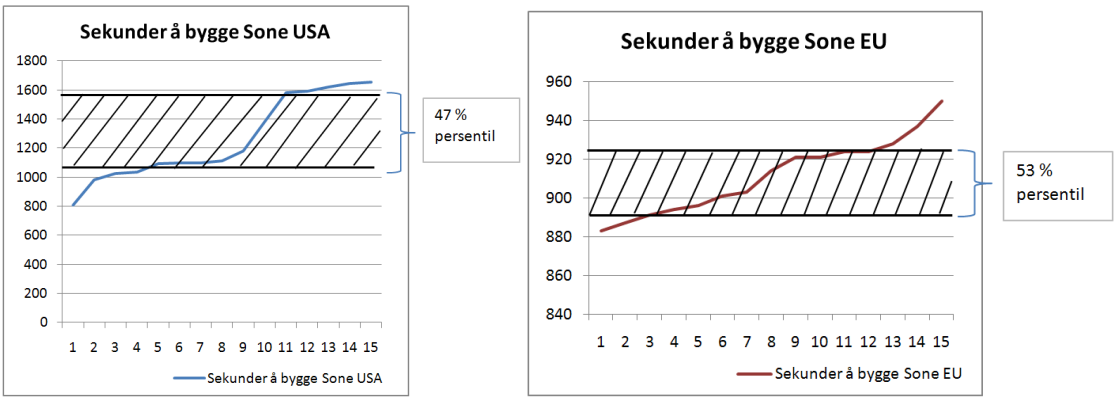
Resultatene av test 1 og 2 (Måle tiden på EC2 US og EU, fra start til stopp på en grafikkprosesseringsoppgave med 2 og 4 slavenoder), viser at Ymer fungerer godt og virker svært forutsigbar. Det var ingen tester som feilet på grunn av tekniske vanskeligheter, men fordi DrQueue opplyser om gjennomsnittstid brukt per bilde, ville vi kontrollere tiden DrQueue oppga som estimert brukt tid, ved å multiplisere tid brukt per bilde med det totale antall bilder. Tiden vi kom frem til her var alltid noe høyere enn tiden DrQueue oppga, noe som antageligvis henger sammen med at DrQueue runder av sine tall nedover mot kun to målepunkter per minutt. Grafikkprosesseringstiden renderfarmene brukte, varierte ikke mye. Dette tilsier at Amazon EC2 ikke har store variasjoner i hvor god ytelse de klarer levere. Resultatene viser også at det er noe forskjell på hvor stabilt tidsbruk det er i de to regionene det ble testet mot. Region EU

har jevnt over et lavere tidsbruk per grafikkprosessering, uavhengig av tidspunkt og størrelse på renderfarm. Region US har i tillegg større variasjoner i tidsbruk enn region EU, dermed kan det tyde på at EU ikke bare leverer bedre ytelse, men også en mer stabil sky. Figur 4 viser at region EU har en jevnere og bedre ytelse enn region US.



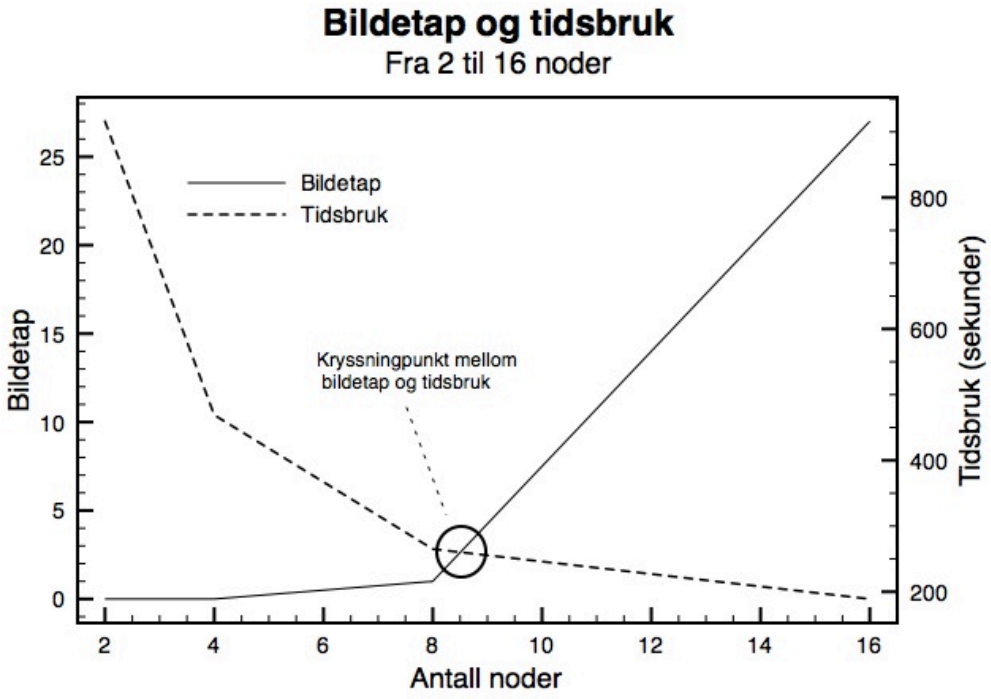
Figur 4

Test 3 (Måle tiden det tar å bygge, starte og pinge et prosjekt på sky) viser at byggetiden er forutsigbar i den forstand at Ymer ikke feiler. Tidsbruken har derimot variert en del, spesielt mot region US. Her var det relativt stor variasjon, sammenlignet med region EU som hadde et mye jevnere tidsbruk. Byggetiden mot region EU var også her en god del kortere enn mot region US, og region US kom heller aldri ned i like lave byggetider som region EU. Om man ser på figur 5, så ser man at region EU er noe mer forutsigbar enn region US. Byggetid mot region EU er 15 min i gjennomsnitt, mens bygging mot region US bruker 19 min i gjennomsnitt.



Figur 5 Persentiler for byggetider mot region US og EU

Test 4 (Måle tiden på en stor grafikkprosesseringsoppgave med ulike antall slavenoder, sone US) viser at vi får bortimot 100 % redusert tidsbruk pr grafikkprosessering for hver gang vi fordobler antall slavenoder frem til renderfarm med 8 slavenoder. Etter dette så ser vi en utflatning i ytelse, samtidig som bildetapet øker markant. Årsaken til dette antas å være kommunikasjonsproblemer relatert til køhåndteringsprogrammet. DrQueue prosesserer dessverre ikke de tapte bildene på nytt, og dette er et problem når man opplever bildetap, ettersom det medvirker til å redusere kvaliteten på resultatet. DrQueue ble valgt fordi den innehar en åpen kildekode, den virket vel utprøvd og høstet mye lovord på internett. Vi planla ikke å sammenligne køhåndteringsprogrammer i prosjektperioden, men man kan man anta at det vil være mulig å optimalisere løsningen ved å for eksempel partisjonere arbeidet i større deler. Dette gjør at det kan være interessant å undersøke andre type køhåndteringsprogrammer i fremtiden. Slik Ymer fungerer nå, så kan det se ut til at man oppnår et optimalt forhold mellom ytelse og kvalitet ved brukt av renderfarm med omkring 10 slavenoder (se figur 6). Etter dette vil det mest sannsynlig ikke være lønnsomt å øke antall noder. Man vil alltids kunne redusere tidsbruken noe med å ha for eksempel 16 slavenoder, men kvaliteten på animasjonen vil da ikke være like god, og man vil også få en større kostnad. Med kvalitet så siktes det til bildetap. Et hvert bildetap vil bety redusert kvalitet og ekstraarbeid for brukeren, de må prosessere disse bildene på nytt.



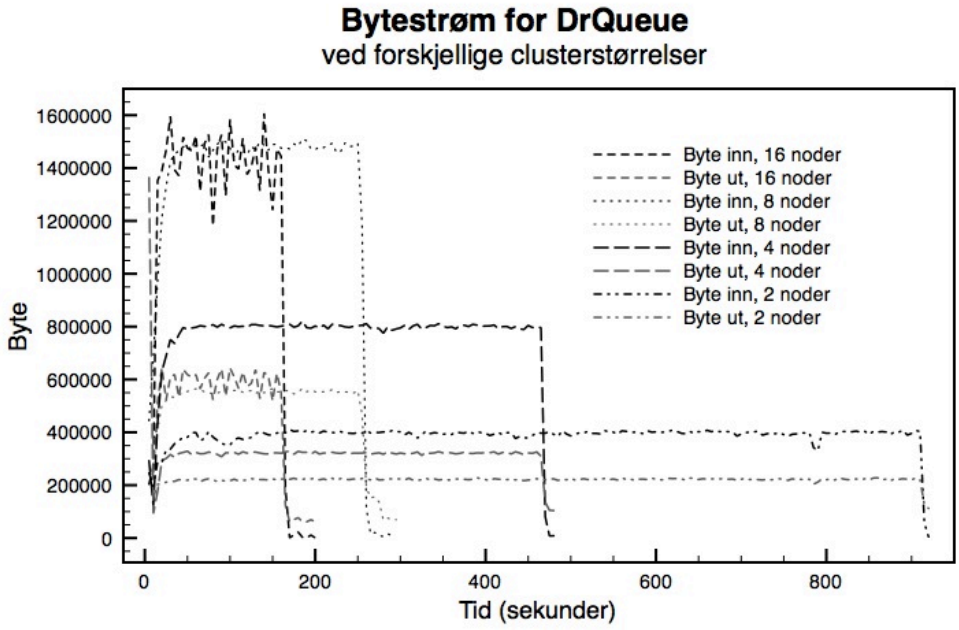
Figur 6

Test 5, (Måle nedlastningstid av en stor fil fra node på sky, til server) viser at nedlastingstiden fra node på sky, til server, er veldig jevn. Det tok i gjennomsnitt 54 minutter å laste ned filen, og variasjonene var små, med en forskjell på 5 minutter mellom laveste og høyeste tid målt.

Test 6 (Måling av trafikkmengde) viser at det er en tydelig grense for hvor mye trafikk Ymer klarer å håndtere. Vi ser at det ikke gir noen stor gevinst å øke fra 8 til 16 slavenoder i en renderfarm. Man sparer riktignok tid, men man får også en pakke og

bytestrøm som er svært ujevn, noe som vi antar vil gi bildetap. Vi antar at DrQueue er hovedårsaken til at det ikke skalerer perfekt med renderfarm med 16 slavenoder.

Som man ser i figur 7, så ligger verken bytes inn eller ut hos DrQueue, noe høyere med 16 slavenoder, sammenlignet med 8. Resultatene av målinger av byte og pakkestrøm viser omtrent det samme, både for NFS og DrQueue. Datatrafikken er derfor kun jevn i forhold til skaleringen, ved bruk av renderfarmer med 2, 4 og 8 slavenoder.



Figur 7

Test 7 (Tid det tar å starte et prosjekt til alle noder er oppe), viser at det er en jevn økning i tidsbruk jo flere slavenoder renderfarmen innehar.

Kostnadsaspektet
For å kunne skape et godt vurderingsgrunnlag for hva tilsvarende maskinvare vil koste i forhold til EC2 instanser, har man med utgangspunkt i den tekniske spesifikasjonen til EC2 instansen c1.xlarge, sett på prisene for tilsvarende maskinvare. Denne instanstypen består av følgende tekniske spesifikasjon:

7 GB minne
20 EC2 Dataenheter (8 virtuelle kjerner, E5410 klokkehastighet på 2.33GHz)
1690 GB lagrings plass

Med den overnevnte instansen som utgangspunkt, ble det gjort et prisoverslag for hva et tilsvarende serveroppsett ville koste ved innkjøp. Alle prisene i denne kostnadsundersøkelsen, er innhentet fra et ledende selskap innen salg av maskinvare. Figur 8 viser hva innkjøpsprisen blir på reell maskinvare med datakraft tilsvarende den datakraften som renderfarmer med 2, 4, 6 og 16 slavenoder har. Prisene inkluderer også en mesternode per renderfarm.

Antall slavenoder	Innkjøpskostnad
2	175696
4	261594
8	450989
16	829778

Figur 8 Innkjøpspris for maskinvare

Driftkostnader knyttet til innkjøp av server

Kostnadene knyttet drift er delt inn i tre hovedkategorier. Disse er:

- Installasjon
  I de aller fleste tilfeller vil det være behov for at en systemadministrator/fagperson deltar ved implementering av systemet. Veiledende pris for dette er estimert til kr 1 100 per time.
- Teknisk drift
  Veiledende pris for en ansatt vil ligge på kr 1 100 per time. Et vanlig timeantall for en deltidsansatt er estimert til 75 timer per måned. Dette tilsvarer 900 timer per år.
- Strømforbruk
  I prisoverslaget presentert nedenfor, vil strømforbruket tilsvare 300W per rakkenhet. Det estimeres med en strømkostnad på 65 øre/KWh, og en oppetid på server som tilsvarer 8 timer per dag. Vi regner med 200W for kjøling av rakktårn, og at serveren har en oppetid på 1 800 timer per år.

Resultater knyttet til estimert teknisk drift

Med utgangspunkt i spesifikasjonene gitt for installasjon, teknisk drift og strømforbruk, får vi følgende kostnadsestimat for teknisk drift:

- Installasjonskostnad, 24 timer, med installasjonstekniker, kr 26 400
- Teknisk drift, kr 990 000.
- Strømkostnader årlig med en oppetid på 1800 timer.

Figur 9 viser hvilke strømkostnader renderfarmer av forskjellige størrelser ville hatt dersom disse representerte maskinvare:

Antall noder	Strøm forbruk i W	Forbruk i KW	Kostnad kr/KWh kr	Årlig basis i KW/h	Kostnad år kr
3 (2 Slavenoder)	1100	1,1	0,65	1980	1287
5 (4 Slavenoder)	1700	1,7	0,65	3060	1989
9 (8 Slavenoder)	2900	2,9	0,65	5220	3393
17 (16 Slavenoder)	5300	5,3	0,65	9540	6201

Figur 9 Strømkostnader

Figur 10 viser kostnadene relatert til drift og oppstart for det første året. I denne tabellen er det også tatt utgangspunkt i ulike renderfarmer, og hva kostnaden ville vært for maskinvare med tilsvarende størrelse:

Antall slavenodernoder	Innkjøp i kr	Oppsett i kr	Drift 50% i kr	Strøm i kr	Total kostnad kr
2	175696	26400	990000	1287	1193383
4	261594	26400	990000	1989	1279983
8	450989	26400	990000	3393	1470782
16	829778	26400	990000	6201	1852379

Figur 10 Driftskostnader første år

Kostnad for oppsett og bruk av Amazon EC2
Pris per time, per instans, altså per node, er kr 3,95 (0,68 USD x 5,8). Dette vil medføre følgende prisøkning ved økning av antall slavenoder:

Antall slavenoder	Pris/h NOK
2	11,83
4	19,72
8	35,50
16	67,05

Figur 11 Timespris pr instans

I prisene vist i figur 11 er mesternoden tatt med i beregningen. Dette innebærer at en renderfarm som har to slavenoder, totalt operer med tre instanser. I figur 12 kan man se sammenheng mellom arbeidstid og pris. Man ser at man kan benytte seg av skyen i mange timer før innkjøp av maskinvare kan bli et lønnsomt alternativ.

Diskusjon og konklusjon

Implementeringen av Ymer hadde en høy teknisk vanskelighetsgrad fordi man ikke hadde noe dokumentasjon på lignende prosjekter som var gjennomført. Skyteknologi er også et relativt nytt felt innen IT, og derfor lite utprøvd. Dette betyr at man i praksis må prøve og feile, og gjøre sine egne erfaringer hele veien. Dette ville betydd store utfordringer for selv en erfaren datakyndig, og som et studentprosjekt så var det naturlig nok ekstra utfordrende.
Ymer fungerte også svært bra i form av stabilitet og forutsigbarhet. Testene som ble utført med renderfarmer hvor nodene var av størrelse c1.xlarge viste også at Ymer hadde begrensninger ettersom ytelsen ikke skalerte jevnt under alle testene som ble utført. Vi prøvde derfor på grunnlag av testresultatene å finne ut hvordan man kunne klare å utnytte ressursene til Amazon EC2 mer effektivt ved å prøve oss frem med forskjellige antall slavenoder. Resultatene av disse eksperimentene viste at ytelseskruven for alvor begynte å flate ut ved bruk av flere enn 10 slavenoder av instanstype c1.xlarge. Man kan anta at det er DrQueue som er hovedårsaken til at vi ikke klarer få en bedre skalering med 16 slavenoder, ettersom vi så at DrQueue tydelig hadde vansker med å håndtere 16 slavenoder. Brukergrensesnittet hang og viste verdier som ikke stemte, og det var vanskelig å administrere renderfarmen. Resultatene av testene som målte bytestrøm og pakkeflyt bekreftet våre antagelser i stor grad, men om årsaken til utflatningen ligger her alene, eller om den ligger i maskinvarens begrensninger eller andre steder, vet vi ikke. Testresultatene viste også at det er en forskjell på regionene Amazon EC2 opererer med. Vi testet om Amazon EC2 ga oss lik ytelse til en hver tid av døgnet, og her så vi at det er EU som skiller seg ut som det beste

alternativet. Byggetiden var mye jevnere, og selv den laveste verdien region US viste, var aldri så lav som den høyeste verdien til region EU. Region US viser også noe lenger tidsbruk per grafikkprosesseringsoppgave, så vi kan derfor konkludere med at region EU gir lavest og mest stabil byggetid, samt best ytelse på skyen.

Kostnadsundersøkelsene som ble utført, forteller oss at det likevel ikke bare er fordeler ved å bruke Amazon EC2, region EU. Prisene man opererer med i Europa er generelt høyere, og dette er et viktig aspekt når man velger region for grafikkprosessering med Ymer. Konklusjonen her er at region US er vinneren om man skal se på pris alene. Et av hovedargumentene for å bruke Ymer, er at det vil kunne gi en bedrift mulighet til å utføre krevende grafikkprosesseringsoppgaver, uten å måtte ut med store investeringssummer i maskinvare. Kostnadsanalysen forteller oss at man med Ymer kan grafikkprosessere på sky i ca 150 000 timer før det blir lønnsomt å kjøpe inn egen maskinvare, forutsett at man ikke bruker over 16 slavenoder av størrelse c1.xlarge.

Ved å lage Ymer har vi gjort skybaserte renderfarmer mer tilgjengelig. I stedet for å sette opp et lignende system, kan man nå laste ned filsystemene sammen med MLN og eksempler på konfigurasjoner. Man kan dermed lett bygge og administrere tilsvarende løsninger selv. Filsystemer samt brukerdokumentasjon kan man finne på MLNs hjemmeside [14].

Fremtidigarbeid

Slik Ymer fremstår i dag, så vil det kreve en del kompetanse å sette den opp, og noe kompetanse å bruke den. Vi har derfor nå i ettertid av skoleprosjektet gått i gang med å lage et skall som syr sammen de forskjellige komponentene, slik at brukeren vil oppleve Ymer som en sømløs og brukervennlig programvare. For at brukeren skal slippe og gjøre serverinstallasjon og drift selv, så har vi valgt å gjøre Ymer til en tjeneste hvor brukeren kan logge seg på en ekstern server hos oss, og derfra administrere renderfarmene sine fra et enkelt brukergrensesnitt. Dette vil gjøre Ymer mer attraktiv, og dermed også mer aktuell for små bedrifter hvor man for eksempel ikke har en egen IT-ansvarlig eller råd til kostnadene det er å investere i dyr og kraftig maskinvare.
Et annet spørsmål er om det er mulig å forbedre prototypen slik at man det blir mulig å utnytte Amazon EC2s ressurser i større grad. Vi antar at køhåndteringsprogrammet vi har brukt er noe av årsaken til at vi ikke er helt der i dag. Dette betyr at det i teorien burde være mulig å lage eller benytte seg av et køhåndteringsprogram som gjør at man klarer håndtere mer CPU-kraft og flere slavenoder. Man bør også undersøke om man kan endre arkitekturen, ved for eksempel å bruke 2 mesternoder. Dette vil kunne gjøre bruksområdet til Ymer en del større, og kanskje den også kan brukes til andre tunge utregningsoppgaver?

Takk

En stor takk rettes til Liv Karin Sameien, Patrik Nylen og Magnus B Sheehan, for deres deltakelse i prosjektet, våren 2010.

Referanser

1. Thain D, Tannenbaum T and Livny M. Distributed computing in practice: the Condor experience. Concurrency - Practice and Experience 17, 2-4 (2005), 323–356

2. McClure S and Wheeler R. Mosix: How Linux clusters solve real-world problems. In USENIX Annual Technical Confer- ence, FREENIX Track (2000), pp. 49–56.

3. Werner Vogels. HPC.NET - are CLI-based Virtual Machines Suitable for High Perfor- mance Computing? SC ’03: Proceedings of the 2003 ACM/IEEE conference on Super- computing, page 36, 2003. IEEE Computer Society.

4. AMAZON WEB SERVICES. http://aws.amazon.com/ec2. EC2 elastic compute cloud.

5. Douceur et al. The utility coprocessor: Massively parallel computation from the coffee shop. USENIX Annual Technical Confenference (ATC’ 10) (2010)

6. Lagar-Cavilla et al. SnowFlock: rapid virtual machine cloning for cloud computing. EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems ACM, (2009)

7. Costa et al. OddCI: on-demand distributed computing infrastructure. MTAGS '09: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (2009)

8. Begnum, K. Managing large networks of virtual machines. Proceedings of the 20th Large Installation System Administration Conference , USENIX, (2006) pp. 205-214

9. Begnum, K. Simplified cloud-oriented virtual machine management with MLN. The Journal of Supercomputing (2010), Springer, ISSN: 0920-8542

10. Barham P, Xen and the Art of Virtualization. SOSP 03, ACM, (2003)

11. Chandra,, R, Zeldovich, N, Chow, J, Lam, M. Virtual Appliances for Deploying and Maintaining Software. Proceedings of the 17th Large Installation System Administration Conference (2003), USENIX, p 181-194

12. Pawlowski B, Noveck N, Robinson D, Thurlow R. NFS version 4 protocol, Proceedings of the 2nd International System Administration and Networking Conference (SANE 2000), USENIX

13. DrQueue the Distributed Render Queue Manager. http://www.drqueue.org/.

14. Blender 3D Modeling Suite. http://blender.org/.

15. MLN homepage. http://mln.sourceforge.net